Out: January 8, 2016
In: January 19-25, 2016

NTNU – Trondheim
Norwegian University of Science and Technology

Exercise 1
TTK4130 Modeling and Simulation

Problem 1 (Modeling, linearization)

An iron ball of radius R and mass m is lifted by a magnet with a coil of N turns and a current i around a core of length l_c and cross section $A = \pi R^2$. The vertical position of the ball is z, which is positive in the downwards direction. The flux ϕ flows through the iron core, then over the airgap, through the ball, and finally along the return path through the open air as shown in Figure 1. The magnetomotive force on the ball is Ni, which can be expressed as

$$Ni = \phi\left(\mathcal{R}_a + \mathcal{R}_c + \mathcal{R}_b + \mathcal{R}_r\right) \tag{1}$$

where

$$\mathcal{R}_a = \frac{z}{A\mu_0} \tag{2}$$

is the reluctance of the airgap, and $\mathcal{R}_c$, $\mathcal{R}_b$ and $\mathcal{R}_r$ are the reluctances of the core, ball and return path, respectively. The reluctances $\mathcal{R}_c$ and $\mathcal{R}_b$ are negligible, and $\mathcal{R}_r$ may be assumed to be constant as the total return path will not change significantly as the ball moves.

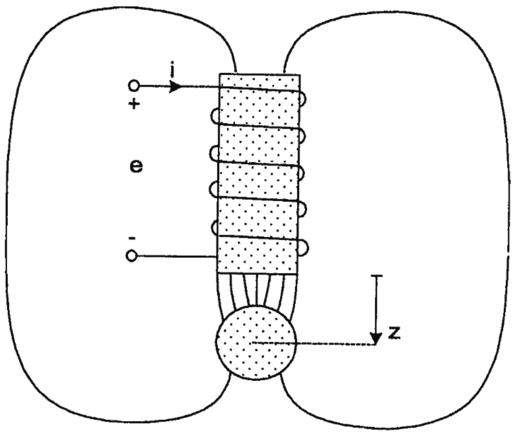


Figure 1: Magnetic levitation experiment

(a) Let the length of the return path be denoted z_0 (assumed constant), and assume a relationship such as (2) for $\mathcal{R}_r$. Write up the total magnetomotive force Ni.

Based on the above, we can calculate the inductance

$$L\left(z\right) = \frac{N\phi}{i} = \frac{N^2 A\mu_0}{z + z_0}. \tag{5}$$

From this, the magnetic force on the ball can be found from

$$F = \frac{i^2}{2}\frac{\partial L\left(z\right)}{\partial z}. \tag{6}$$

(b) Use Newton's second law to find the equation of motion for the ball.

(c) Linearize about a constant position z_d (and a corresponding constant current input, i_d).

Problem 2 (Network modelling of motor with two elastic loads using Simulink)

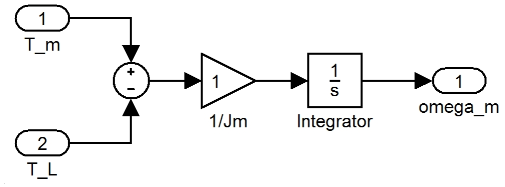
In this problem, we will attempt to use a "network modeling" approach in Simulink, that is, try to use physically motivated model interfaces, even though Simulink has no built-in mechanisms to support this[1]. The system we will model is a rotary motor with two elastic loads, that is, a mechanical system which is natural to divide into three parts.

A rotary motor has some device for setting up a motor torque T_m on a rotary shaft that rotates with angular velocity ω_m. The equation of motion for the shaft is

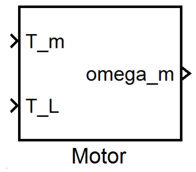
$$J_m \dot{\omega}_m = T_m - T_L,$$

where T_L is the load torque acting on the shaft. Assume the inertia is $J_m = 1\text{kg} \cdot \text{m}^2$.

(a) Implement the motor in Simulink as illustrated in Figure 2. Choose yourself if you want to hardcode the parameter J_m (as in the Figure), or you want to make it a mask parameter.



(a) Under the mask



(b) The mask

Figure 2: Simple motor implemented in Simulink

From an energy-flow (network) point of view it is the power delivered to the motor that makes the motor run. This power is $P = T_m \omega_m$, and from this perspective natural inputs are T_m and ω_m. Similarly, the power delivered from the motor to a load is $P = T_L \omega_m$, which means natural outputs from the motor model is T_L and ω_m. However, in block-oriented (signal-flow oriented) tools like Simulink which has *unilateral interconnections*, the choice of inputs and outputs must be based on the way the model is solved/implemented computationally.

We will extend the model with a number of elastic loads, each with the following model (see Section 1.4.4 in the book):

$$J_i \dot{\omega}_i = T_{i-1} - T_i \tag{11a}$$

$$\dot{\theta}_e = \omega_{i-1} - \omega_i \tag{11b}$$

$$T_{i-1} = D_i \left(\omega_{i-1} - \omega_i\right) + K_i \theta_e \tag{11c}$$

where θ_e is the difference in rotor angles between the driving rotor and the elastic load rotor, T_{i-1} and ω_{i-1} are the torque and rotational speed on the driving rotor, and T_i and ω_i are the torque and rotational speed on the elastic load rotor.

(b) Based on the equations, what are natural signal-flow (computational) inputs and outputs, and why? (Hint: See next question.) What are natural energy-flow inputs and outputs?

(c) Implement a generic elastic load as a Simulink sub-system, as shown in Figure 3. Either hardcode $J = 1\text{kg} \cdot \text{m}^2$, $K = 0.5\text{kg} \cdot \text{m}^2/\text{s}^2$ and $D = 0.01\text{kg} \cdot \text{m}^2/\text{s}$, or use mask parameters.

(d) Put together the motor and two elastic loads. The last elastic load should not have an external load connected ($T_i = 0$). Let the motor torque be given as a step, and simulate. Comment on the behavior of the rotational speed of the last load.

[1]Simulink has an extension, Simscape, which has such features, but we will not be using Simscape in this problem.

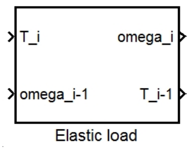


Figure 3: Elastic load Simulink mask

(e) Finally, we will look at the Bode plot from input motor torque to output rotational speed on the last load. We will let Simulink help us (requires Simulink Control Toolbox):

1. Right-click the line you want as input, choose 'Linearization point' and 'Input point'. Do correspondingly for the line you want as output.
2. Then, in the menu, choose 'Tools' → 'Control Design' → 'Linear Analysis'. A new window will appear.
3. (Normally, we would now have to choose an operating point about which to linearize, but in this case the system is linear, so the operating point does not matter.)
4. Choose 'Linearization Task', select 'Bode response plot', and press 'Linearize Model'.

Alternatively (if you don't have Simulink Control Toolbox):

1. Put a Sources/In1-block where you want the input, and a Sources/Out1-block where you want the output.
2. In Matlab, call `[A, B, C, D] = linmod('model');`, if model.mdl is the filename of your model.
3. Plot the bode response: `bode(A, B, C, D);`

Comment on the plot.

Problem 3 (Network modelling of motor with two elastic loads using Dymola/Modelica)
In this problem, we will model the same process as in the previous question using Dymola/Modelica. Instead of implementing the models from scratch (which we will learn how to do later in the course), we will model by using predefined models from the Modelica Standard Library (MSL).

(a) Start Dymola. Choose 'File' → 'New' → 'Model'. Enter the name of the model (for example 'MotorWithElasticLoads'). Let the rest be empty, and press 'OK', and then 'Accept' at the warning that will appear[2]. In the pane at the left hand side, press/unfold 'Modelica' to open the Modelica Standard Library. Open 'Mechanics' → 'Rotational'. Drag and drop 'Sources' → 'Torque', 'Components' → 'Inertia' and 'Components' → 'SpringDamper' to put together the motor with two elastic loads as shown in Figure 7. Use a 'Blocks' → 'Sources' → 'Step' to attach a step input to the torque.

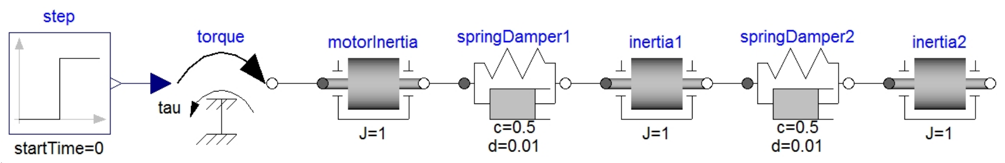


Figure 7: Dymola model of motor with two elastic loads

Press the 'Simulation' tab (choose "Simulation view") at the bottom, choose 'Simulation' → 'Simulate' in the menu. Plot the rotational speed of the last load (by navigating in the left hand side

[2]Normally, all models will be part of a package of models, thus the warning.

pane, the symbol w is used for rotational speed), and compare to the previous problem. Also comment on the amount of information in the graphical view of the model (Figure 7) with the Simulink version of the overall model.

(b) Identify what variables the Modelica Standard Library uses to connect rotating mechanical systems. Do this by going back to the "Modeling view" (press 'Modeling' tab), and open 'Mechanics' → 'Rotational' → 'Interfaces' → 'Flange_a'. Press the documentation icon (the big I) in the menu bar. Compare with the previous problem.

(If you want: Dig into the models to check that the equations used are the same as in (11). This is not expected, we will learn more about the Modelica language and the Dymola program later.)

(c) Make a Bode-plot of the model, by

1. Add inputs and outputs to the model: Remove the Step-block, and connect a 'Mechanics' → 'Rotational' → 'Sensors' → 'SpeedSensor' to the last load. Add 'Blocks' → 'Interfaces' → 'RealInput' to the motor torque, and 'RealOutput' to the sensor output.
2. Go to the Simulation-view. Choose 'Simulation' → 'Linearize'.

Dymola will now export a linear(ized) model to a binary Matlab-file, called dslin.mat, in the directory where you have saved your model. Open matlab to import this and make a Bode plot, for example by using:

```
% load output from Dymola linearize
load dslin
% ABCD is A, B, C and D matrix stacked into one matrix
% nx is number of states (dimension of the A matrix)

A = ABCD(1:nx,1:nx); B = ABCD(1:nx,nx+1:end);
C = ABCD(nx+1:end,1:nx); D = ABCD(nx+1:end,nx+1:end);

% Plot Bode response
bode(A,B,C,D)
```

Compare with the Bode plot in the previous problem.